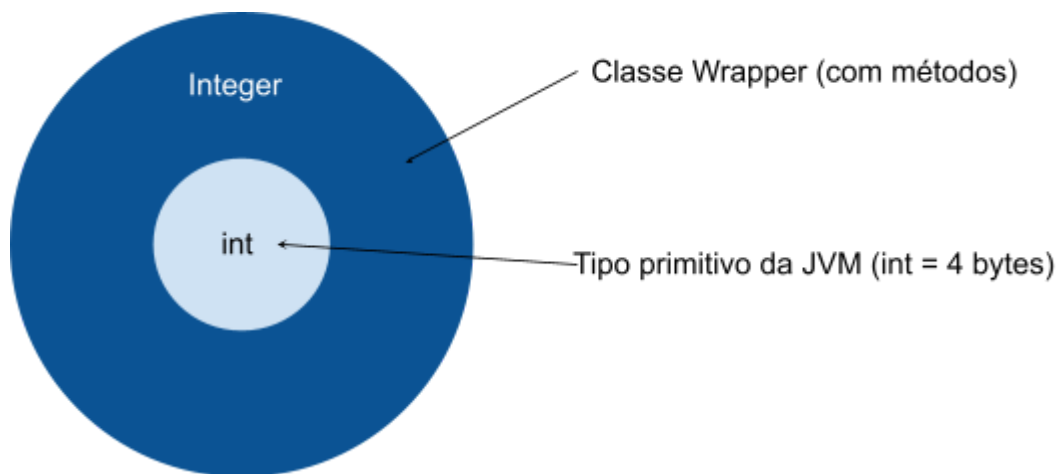


## 1. Na tecnologia Java, defina o que são e para que servem as Classes Wrappers.

Wrappers são classes que encapsulam os tipos primitivos (provenientes da JVM). Cada classe wrapper representa um tipo primitivo e possui métodos que auxiliam o uso desses valores (em atividades como conversão...). As classes wrapper possuem conversão implícita automática com seus tipos primitivos respectivos.



As coleções do Java geralmente utilizam as classes wrapper para definir através de generics qual o tipo armazenado e são capazes de armazenar e manipular múltiplos valores armazenados como Objetos Wrapper.

## 2. De forma sucinta, comente a aplicação das seguintes classes:

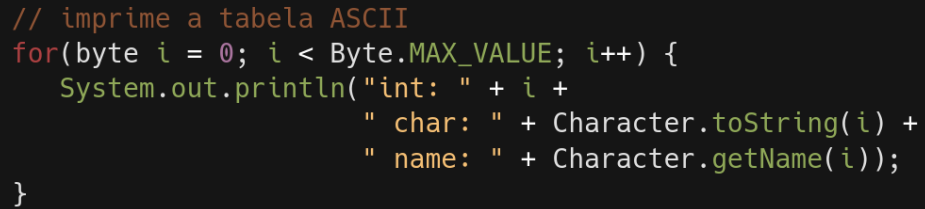
- a. **Integer**: Classe Wrapper que empacota o tipo primitivo `int` que representa valores numéricos inteiros entre -2147483648 e 2147483647.
- b. **Boolean**: Classe Wrapper que empacota o tipo primitivo `boolean` que representa valores booleanos podendo ser `false` (falso) e `true` (verdadeiro).

- c. **Character**: Classe Wrapper que empacota o tipo primitivo char que contém caracteres Unicode entre 'u0000' e 'uFFFF' de 16-bits fixos.
- d. **Double**: Classe Wrapper que empacota o tipo primitivo double que representa valores numéricos decimais entre  $4.9 * 10^{-324}$  e  $1.7 * 10^{308}$  (valores com 15 dígitos após a vírgula).
- e. **Byte**: Classe Wrapper que empacota o tipo primitivo byte que representa valores inteiros entre -128 e 127.
- f. **Short**: Classe Wrapper que empacota o tipo primitivo short que armazena valores inteiros entre -32768 e 32767.
- g. **Float**: Classe Wrapper que empacota o tipo primitivo float que armazena valores decimais entre  $2^{-149}$  e  $(2-2^{-23}) * 2^{127}$ .
- h. **Long**: Classe Wrapper que empacota o tipo primitivo long que representa valores inteiros entre -9223372036854775808 e 9223372036854775807.

**3. Considere as classes citadas no item “2”. Escolha 3 destas classes e construa um código simples que mostre um exemplo de sua aplicação para elas. Copie e cole o código de teste abaixo.**

(página próxima)

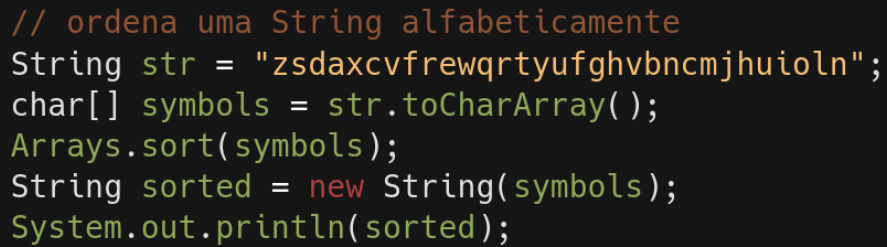
## I. Character



```
// imprime a tabela ASCII
for(byte i = 0; i < Byte.MAX_VALUE; i++) {
    System.out.println("int: " + i +
        " char: " + Character.toString(i) +
        " name: " + Character.getName(i));
}
```

```
for(byte i=0; i<Byte.MAX_VALUE; i++) {
    System.out.println("int: " + i +
        " char: " + Character.toString(i) +
        " name: "+Character.getName(i));
}
```

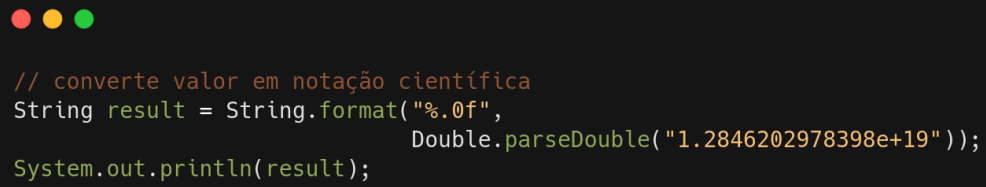
## II. String



```
// ordena uma String alfabeticamente
String str = "zsdaxcvfrewqrtyufghvbncmjhuioln";
char[] symbols = str.toCharArray();
Arrays.sort(symbols);
String sorted = new String(symbols);
System.out.println(sorted);
```

```
String str = "zsdaxcvfrewqrtyufghvbncmjhuioln";
char[] symbols = str.toCharArray();
Arrays.sort(symbols);
String sorted = new String(symbols);
System.out.println(sorted);
```

### III. Double



```
// converte valor em notação científica
String result = String.format("%.0f",
                               Double.parseDouble("1.2846202978398e+19"));
System.out.println(result);
```

```
String result = String.format("%.0f",
                               Double.parseDouble("1.2846202978398e+19"));
System.out.println(result);
```